DIGITGRAPH
P. O. Box 4249
Tucson, Arizona 85717

RSTS-11 SIG
DECUS

FEBRUARY 1978 RSTS-11 SPECIAL INTEREST GROUP NEWSLETTER VOL 5 NO 1

FROM THE SIG CHAIRMAN

Those of you who weren't among the 500+ RSTS users at Fall DECUS in San Diego, missed a very useful and constructive meeting. Version 6C of RSTS was announced for release in early 1978 and DEC assured the users of continued commitment to RSTS and BASIC-PLUS. The development commitment doesn't go as far as some of us old-time users feel it should, but it still was a positive response.

The first user reaction from the field came in on BASIC-PLUS II. The reaction, as I perceived it, was that BASIC-PLUS II does some good things such as improving execution speed and allowing larger program sizes, but still has a way to go before it obtains its design objectives. DEC has already identified several areas that need improvement and they welcomed user input. DEC emphasized that BASIC-PLUS II is not meant to be a replacement for BASIC-PLUS. It is instead a language that will eventually allow a growth path to larger DEC computers. It is my feeling BASIC-PLUS II has its place in the sun, but I think it is a shame the new RSTS users are being pushed into it by their sales people before it is really solid and without getting them started on BASIC-PLUS.

The Symposium had a major difference this time--no Dian Pekin. The woman, who had proved that it is possible to have the same RSTS Product Manager at three DECUS meetings, has moved to bigger and better things. There is no way to adequately thank Dian for all the help, time, and work she always willingly gave the RSTS SIG. Her last act as RSTS Manager was to have stickers made up that said "Share RSTS with a Friend". I'd like to thank Dian for sharing her "time" and her "friendship" with the RSTS SIG.

Another change that happened at DECUS is that Bill Sconce has been appointed to the position of Symposium Coordinator on the DECUS board. I did twist Bills' arm and ask him to continue working with the SIG. He said he still has wishes and will continue working on the Wishlist.

Bob Branton
RSTS SIG Chairman
3115 Brincrest Drive
Dallas, TX 75234
214-243-0420

DECUS and DIGITAL EQUIPMENT CORPORATION assume no responsibility or liability whatsoever for any articles or other information appearing in this document.

Here is a list of the hard working people who are chairing RSTS Subgroups:

A. COMMERCIAL RSTS USERS
Stu Katz
Commercial Computer Services
1011 E. Tonky
Des Plaines, IL 60018
312-298-7766

EDUCATIONAL RSTS USERS
Howard Kipple
New Mexico Junior College
Hobbs, NM 88240
505-392-6526

CUSP EDITOR
Jon Spencer
PASCAL Computing
432 S. Barrington Ave.
Los Angeles, CA 90049

JSTATS, DSTATS DOCUMENTATION
Mike Mayfield
Data Processing Design, Inc.
6980 Aragon Circle, #7
Buena Park, CA 90620
714-994-4971

NEWSLETTER EDITOR
Murray Stinson
Scientific Placement, Inc.
2700 S. Post Oak Rd., #2430
Houston, TX 77056
713-621-9811

LARGE RSTS SYSTEMS GROUPS
Joan Salkin
U. S. Leasing
633 Battery St.
San Francisco, CA 94111
415-445-7400

TECHNICAL EDITOR
Carl Marbach
Acme-Hardesty Company
910 Fox Pavilion
Jenkintown, PA 19046
215-886-0200

B. WORKING GROUPS:

RSTS USER PROFILES
Bill Tabor
Florida Computer, Inc.
99 N. W. 183 St.
N. Miami, FL 33169
305-652-1710

RSTS DECUS LIBRARY PROGRAM
Ardoth Wilson
Central State University
100 N. University
Edmond, OK 73034
405-341-2980 Ext. 321

PROGRAMMING STANDARDS DEVELOPMENT
Roy Swatzell, Jr.
V. A. Hospital
Room 2 R 12
Birmingham, AL 35233
205-933-8103 Ext. 239

RSTS USER WISH LIST DEVELOPMENT
Bill Sconce
Industrial Specialties
P. O. Box 4228
Louisville, KY 40204
502-895-2914

Volunteers Wanted
for Spring DECUS Seminar

RSTS is becoming very complex and intricate. Each of us is gaining specialized knowledge in some areas: we need to find good ways to share our knowledge.

At Fall DECUS we tried the experiment of a new users' session, in which experienced users were simply available and present for questions. There will be a similar meeting, at Spring DECUS, in which experienced users will be present for one-on-one questions and discussions. Have you ever wished you could talk to another user about those tricky BASIC-PLUS/FORTRAN/RT11/RSX/RMS/... (the list goes on and on) questions? This is your chance! Or do you already know some of the pitfalls and shortcuts? Come tell us!

If you have experience in some special area please send me your name. We don't need geniuses -- we need people willing to share their knowledge even if it's incomplete! Some names are already in (see below). Some possible areas of experience: BASIC-PLUS2, MACRO, RMS, RSX, RT11, COBOL, PASCAL, optimization, error recovery, FORTRAN, TKB, statistics...send your name, phone, address and specialty to Mike Dash, John Fluke Mfg. Co., P.O. Box 43210, Mountlake Terrace, WA. 98043.

Bob Liquori	Nassau Boces DP, NY	large systems
John Crossman	Borden Chemical, Ohio	large systems
Ron Webster	University of Toledo, Ohio	FORTRAN,BASIC
Brian Lockrey	Arizona State University	FORTRAN,BASIC COBOL,DYNPRI
Bruce Gaarder	Macalester College, Minn	Sysgen options, stats, system tuning
Bill Boyle	MLPI, Arizona	RSTS internals
Dave Mallery	Nationwide, Pa	System tuning

January 9, 1978

Open letter to the RSTS SIG-

The RSTS SIG is the most beneficial method of information exchange available to the RSTS world. The SIG has basically two avenues of communication available. The first being the newsletter and the second being the semi-annual symposium. The newsletter is a good method of communication but as anyone that has attended a symposium knows, the opportunity to talk in both formal and informal discussions with fellow RSTS users and with the RSTS developement team can not be beaten.

This interchange between users needs to be expanded in order to further cover the multitude of problems that RSTS now covers. With multiple languages and new monitor executive changes being introduced we need user input to assist in reaching sound decisions that each of us must make.

In this regard we are calling upon each of you to consider what information you might have that would be beneficial to the rest of the RSTS world. You might have a particular application that would be of interest or you may use a set of particular modules that would be useful for the rest of us to hear about. Remember that if each of us pitch in with just a little effort no one becomes burdened with providing a mass of information.

In the same vein, if you have a particular topic you would like to see discussed at the Chicago meeting write a letter or call me and I will try to contact someone to present a session. This includes any topics you would like to see DEC present.

Chicago will have some new interesting twists for the RSTS portion of the symposium. These will include a RSTS "get together room" and terminals tied into various RSTS systems.

Remember, your participation will help further RSTS and will provide you more in return then you can imagine. Join us in Chicago and let us hear from you in the next couple of weeks.

Stuart M. Katz
RSTS SIG Symposium Coordinator

RSTS MARKETPLACE

DESCRIPTION	CONTACT
1.) Subroutines to phonetically code last names/for storage and retrieval via standard phonetic techniques. 2.) Screen generator programs for the Tektronix 4023 buffered CRT terminal. 3.) Complete letter writing system using paragraph/letter files to build custom upper/lower case letters from form paragraphs. System allows insertions as well as a user composed paragraph.	Murray Stinson Scientific Placement, Inc. 2700 South Post Oak, #2430 Houston, Texas 77056 713-621-9811 4/1/77
1.) DATA BOSS/2 - Data Base Management Systems and Applications Monitor. Conversational setup, an English-like Inquiry Language, fast retrieval of millions of records via up to 65 keys. DBMS provides device independence and enables I/O with any ASCII file or device.	John H. Wright Florida Computer Inc. 99 N. W. 183 Street North Miami, Florida 33169 305-652-1710 4/1/77
"FIPS" - Financial Information Processing System. An easy to use computerized accounting system designed by accountants for accountants. Includes general ledger, payroll, payables, cost accounting, financial statements, etc. Features: ● Automatic Interface to General Ledger ● Complete Audit Trail ● Unique Report Formatter ● Company Consolidation and Overhead Budget Analysis	Tim Schroeder or Mike Niernberg Environmental Developers, Inc. 13693 East Iliff Avenue Denver, Colorado 80232 303-758-4619 4/1/77
The Statistical Programs for the Social Sciences/Carleton (SPSS/C) for the PDP-11. It is written in FORTRAN IV and tested under DOS V9, RT-11, and RSTS.	Graham Kimble or Carl Henry Computer Center, Carleton College Northfield, Minnesota 55057 507-645-4431 Ext. 204, 504 4/1/77
RSTS/E OMSI SORT-1 - Fast SORT for Large Files. Machine language SORT Package compatible with DEC BASIC-Plus SORT runs up to 50 times faster than DEC version. Sorts files containing over one million records. Very attractive price with OEM and multiple CPU discounts. Runs on version 6B as normal 16k word user job without software or hardware modifications. Chain out and back or run stand-alone.	OMSI Computing 4015 S. W. Canyon Road Portland, Oregon 97221 503-248-5900 4/1/77
PDP-11 TO /370 2-WAY DATA TRANSFER - Two programs: 1.) RSTS/E V6-B Basic-Plus program to transmit/receive data files to/from IBM/370-158, using BINARY I/O, on a full-duplex line. 2.) IBM VS1 ASSEMBLER program, using BTAM, may be called by COBOL or other /370 languages, to send, receive, rename, or kill files on any PDP-11, connected by dedicated full-duplex line. This system is currently in use in our installation to transmit data files in both directions between our /370-158 and a PDP-11/34 which is 200 miles away, connected by a dedicated line, as well as to our PDP-11/70, which is in the same room, and is connected by another dedicated line. Current transmission rate is 1200 Baud.	I. F. (Bud) Dawson MacMillan Bloedel Limited 1075 West Georgia Street Vancouver, B.C. Canada V6E 3R9 7/1/77

DESCRIPTION	CONTACT
HASPBOX is a communications front-end for RSTS/E. It allows full-duplex data transmission between RSTS and IBM/360/370 mainframes via HASP protocall. The system allows up to 4 input and 4 output streams to be operating concurrently, and also supports a HASP operator's console. The system has hooks into the standard RSTS batch processor and line printer spooler combined with a input Queue system. RSTS-LINK is a communication link for use between one or more RSTS based PDP-11 systems. The link provides a fast, low-cost link between the systems allowing transfer rates of up to 75KB. The link is fast enough for use in a real-time application such as data-base inquiry. The system has hooks into the standard RSTS batch processor and line printer spooler combined with a input Queue system. This link creates a true remote-batch facility for each interconnected RSTS system.	Gerry Moersdorf Software Results Corp. 455 West Seventh Ave. Columbus, Ohio 43201 614-421-2094 10/1/77
BMDP Statistical Programs for the PDP-11. A complete implementation of the February 1976 version of the programs. Twenty-six different programs. Entirely in Fortran and Macro. Developed under RSTS, installation under RSX, LAS, and UNIX is also possible. Distribution includes 2400' tape with complete source code, test data, sample output, and full documentation.	Dr. James Krupp Software Development Inc. Middlebury College P. O. Box 500 Middlebury, Vermont 05753 802-388-7976 10/1/77
1.) TAM, a format-language-driven terminal-handling front-end software module for transaction-processing applications under RSTS/E. 2.) KDSS, a key-to-disk data entry subsystem under RSTS/E. (In both TAM and KDSS, the performance-critical portions are implemented in PDP-11 assembly language and installed under RSTS/E as run-time systems.) 3.) FSORT3, an ultra-fast run-time system capable of sorting up to 16 million blocked records or keys. Record and key formats are compatible with the RSTS-11 Sort package. The FSORT3 package contains modules for replacing SQWIK, MQWIK, and XQWIK. FSORT3 may also be used stand-alone. The package uses about 2% of the CPU time of its BASIC-PLUS counterpart and runs up to 30 times as fast in stand-alone wall time measurements. 4.) INTCOM, a set of BASIC-PLUS programs allowing two RSTS/E systems to communicate, including ASCII file transfer, via telephone line up to 300 baud.	Tom Evans Evans Griffiths and Hart, Inc. 55 Waltham Street Lexington, Massachusetts 02173 10/1/77
RMS-11/BASIC-PLUS 2 Integrated Financial Accounting System designed for Universities, Municipal Government, Government Agencies, and Commercial Users: Modules include: - Purchase Order Module - Accounts Payable Module - General Ledger Module - Budget Module - Vendor Module - Balancing and Audit Module	Dennis Sakamoto or Richard Rock SRB Computer Systems Ltd. 50 Crown St., Suite 175, Saint John, N.B. E2L 2X6, Canada 506-693-0620 2/1/78

DESCRIPTION	CONTACT
WORD-11: Word Processing for RSTS/E. Extremely powerful multi--terminal word processing system. Uses standard VT52 terminals and any printing device, including letter quality daisy wheel printers. Implemented as a runtime system in MACRO-11. Functionally similar to DEC's PDP-8 based word processing system.	Rich Marino or Mike Mayfield Data Processing Design, Inc. 6980 Aragon, Suite 8 Buena Park, CA 90620 714-994-4971 2/1/78
SAVER: Replacement for BACKUP for RSTS/E. Written in MACRO-11, provides fast, reliable disk to tape, and disk to disk backup. Writes PIP-readable files, maintains all file and accounting information; restore operation utilizes optimal UFD/file placement algorithms. Fast, efficient, with user-oriented interface.	Rich Marino or Mike Mayfield Data Processing Design, Inc. 6980 Aragon, Suite 8 Buena Park, CA 90620 714-994-4971 2/1/78
PAGE is a full screen video terminal text editing program designed to provide the user the convenience of cursor controlled text editing. Rather than entering "blind" commands to search, position, and modify text, PAGE displays the text in context on the terminal screen. The user can visually locate text to be modified, then position the cursor under the desired character or line and key in the change. The change is then immediately reflected on the screen. Both programmers and non programmers can easily learn PAGE. PAGE is ideal for producing and editing documentation.	Pam Biederman Interactive Information Systems P.O. Box 37403 Cincinnati, Ohio 45222 513-761-0132 2/1/78
IIS Program Dispatcher: The dispatcher is a facility to allow end users of RSTS/E to execute certain sets of predetermined programs without having to know or understand anything about the RSTS/E monitor or file structure. The user simply enters a "transcation code" which the Dispatcher validates for that user and maps to a program name. The Dispatcher can greatly simplify the task of teaching end users to use the computer, and requires almost no additional system overhead. The Dispatcher can also record system utilization information and can automatically log users back in after a system crash.	Pam Biederman Interactive Information Systems P.O. Box 37403 Cincinnati, Ohio 45222 513-761-0132 2/1/78
IIS Sort/Merge Package: The IIS Sort/Merge Package is a set of five programs designed to perform the functions of sorting and/or merging files. Files may be either record I/O or formatted ASCII and may contain the standard field types (character string, packed integer and floating point or ASCII). Up to ten fields in sixteen files may be sorted in ascending or descending sequence. It is also possible to chain to and from application programs. The critical sorting code is written in MACRO-11 Assembler language which results in an approximate 100-1 improvement in execution efficiency.	Pam Biederman Interactive Information Systems P.O. Box 37403 Cincinnati, Ohio 45222 513-761-0132 2/1/78
IIS Data Base Management System: The IIS Data Base Manager provides users with a comprehensive and powerful data management tool at an affordable cost. The system incorporates many of the facilities found only in larger data base systems at a fraction of the cost, and is designed to be very easy to use. The system provides data base access and update capability simultaneously to a large numbers of on-line users. Each user may access up to 128 files at any one time.	Pam Biederman Interactive Information Systems P.O. Box 37403 Cincinnati, Ohio 45222 513-761-0132 2/1/78

RSTS MARKETPLACE

DESCRIPTION	CONTACT
IIS Data Logger: The Data Logger is a data management facility for sequentially organized information. The Data Logger can store and retrieve an almost unlimited number of files on either disk or magnetic tape, and is provided with two application systems: a Data Recovery system which allows users to "log" on-line updates to critical files and provides for the recovery of these files in the event of a "hard" system crash, and the Transaction Accounting system which records user logins and logouts and can record the execution of individual programs. Facilities are provided for recording user generated information and extracting and printing billing information.	Pam Biederman Interactive Information Systems P.O. Box 3[illegible]403 Cincinnati, Ohio 45222 513-761-0132 2/1/78
MIRCOM: A software package wtitten in BASIC/PLUS to transmit messages or files of any type between two or more computers operating under RSTS/E. - No special communications hardware is needed. The system requires only a single standard serial interface operating at any standard speed on each machine. - A message send/receive protocol is incorporated which allows binary data to be transmitted and received and provides recovery procedures for data transmission errors.	Rex Hill or Gil Jost Diagnostic Computer Division Mallinckrodt Institute of Radiology 510 S. Kingshighway St. Louis, Missouri 63110 314-454-2181 2/1/78

Murray Stinson
Scientific Placement, Inc.
2700 South Post Oak Road
Houston, Texas 77027

Dear Sir:

National Information Systems, Inc., founded in 1972, is a recognized leader in on-line interactive software systems for the DECsystem 10 and 20. Our primary product is a DATA BASE MANAGEMENT SYSTEM called DPL.

DPL (Data Processing Language) is a complete Information Management System which can be used as a query facility and report writer for existing FORTRAN and COBOL files WITHOUT conversion. But DPL is more than just an extensive report writer. It is a rational, real world approach to Relational Data Base Management. DPL provides a concise language to communicate with your data base. It is both simple enough to be used by non-technical personnel, and yet has sufficient capability to effectively process large, complex, integrated data base applications.

DPL's structured programming language encourages the use of modular programming techniques. But more significantly, DPL has built-in models of the common data processing functions, such as updating, validating and report writing. The use of these models will reduce the amount of coding required to implement an application by up to 90%. This substantial gain in efficiency means what would have been weeks of programming effort will now be reduced to days. Yet the operational speed is not sacrificed because DPL programs usually run as fast, if not faster than comparable programs written in other languages.

Because of the extensive use of DPL at more than 30 in-house installations and on ADP, Network Services (a national commercial timesharing service where it is called IPL); we have often been approached to write a version for the PDP-11 under RSTS.

Should any installation be interested in participating in such a sponsership, please contact me as soon as possible at (415) 941-8345. It is our anticipation that we will have the required backers committed by the end of March and translation commencing by April. At this time, it is anticipated the project will take approximately one year to complete with first release expected by the end of 1978.

Sincerely,

NATIONAL INFORMATION SYSTEMS, INC.

John Enyedy

John Enyedy
Western Regional Manager

SAINT MARY'S COLLEGE, Winona, Minnesota 55987

December 22, 1977

Murray Stinson
Scientific Placement, Inc.
2700 S. Post Oak Rd., #2430
Houston, Texas 77056

Dear Sir:

In the RSTS/E release notes January 1977, the format for storing alphanumeric information in FORTRAN was described as, succeeding characters are stored in ascending bytes. This presents a problem to the programmer who would like to use logical comparisons to obtain the ASCII collating sequence on more then one character at a time, as in sorting or searching. The attached subroutine is a method of circumventing the problem. It arranges words such that items read in 'A2', 'A4','A8' format will collate properly.

Yours truely,

Gerry Cichanowski
Director Academic Computing

```
DSWAP   RT-11 MACRO VM02-11   22-DEC-77 10:06:14 PAGE 1


                        .TITLE  DSWAP
                                                ;SWAPS 'N' WORD'S BYTES
                                                ;A WORD AT A TIME.
                                                ;THE FORTRAN CALL IS
                                                ;CALL DSWAP(N,X)
                                                ;WHERE N IS THE NUMBER OF WORDS TO SWAP
                                                ;AND X IS AN ARRAY OR SINGLE VARIABLE
                                                ;WITH AT LEAST N WORDS IN IT.
                        .GLOBL  DSWAP
000000 005725 DSWAP:    TST     (%5)+           ;SKIP # OF ARGUMENTS
000002 013503           MOV     @(%5)+,%3       ;NUMBER OF WORDS TO SWAP
000004 000375 TOP:      SWAB    @(%5)           ;SWAP BYTES
       000000
000010 062715           ADD     #2,(%5)         ;PREPARE FOR NEXT WORD
       000002
000014 077305           SOB     %3,TOP          ;DECREMENT R3 IF > 0 BRANCH TO SWAP NEXT WORD
000016 000207           RTS     %7              ;RETURN
       000001'          .END
```

MeadPaper

Specialty Paper Division
South Lee, Massachusetts 01260

Telephone: 413-243-1231

Mr. Murray Stinson
Scientific Placement
2700 South Post Oak Road
Suite 2430
Houston, TX 77056

Dear Mr. Stinson,

If possible, I would like to place the following notice in the Newsletter.

Our company is a new RSTS user, being currently involved in converting from an IBM System 3/Model 10 to a PDP 11/34. We are using both BASIC-Plus and RPG II and are using the RPGII / RTSLIB file structure, accessed by the BASIC-Plus programs via the RPGFNS functions package supplied by DEC. We would appreciate hearing from any other users who have made or are making a similar conversion. Our main problem has been the lack of speed involved in adding an item to an IAM file (updating the index is causing the problem) and would appreciate hearing any solutions developed by others.

Thanking you in anticipation,

Sincerely yours,

Jerrold S. Crowell
Programmer/Analyst

5522 Big Oak Drive
San Jose, CA 95129
December 17, 1977

Mr. Murray Stenson
RSTS Newsletter Editor
Scientific Placement Inc.
2700 S. Post Oak Rd. #2430
Houston, TX 77056

Dear Mr. Stenson:

Could you publish the following in your next newsletter?

I am interested in sharing ideas on management simulation programs running in a conversational mode on PDP11/34s.

Two programs under evaluation are:
THE BUSINESS MANAGEMENT LABORATORY
Business Publications Inc.

CONVERSATIONAL EXECUTIVE GAME
Queens College of the City University of New York

My address is:
San Jose City College
Mike Ausloos
5522 Big Oak Drive
San Jose, CA 95129

Thanks.

Sincerely,

Mike Ausloos
Instructor

american national standards committees
X3—computers and information processing
X4—office machines and supplies
operating under the procedures of the
American National Standards Institute

doc. no. : X4A12/77-28

date: : August 22, 1977

project :

milestone :

reply to :

The American National Standards subcommittee X4A12, word processing, is in the process of developing standards for keyboards and character sets for document preparation equipment. This equipment is often known as word processors, magnetic keyboards, automatic typewriters.

We need your help. Please complete this survey form and return it to: Patricia M. Caroom
DECUS Standards Coordinator
Box 13047
Austin, Texas 78711

Your cooperation is sincerely appreciated.

E. H. Kight/WFH

Edward H. Kight
Chairman, X4A12

secretariat: Computer and Business Equipment Manufacturers Association
1828 L Street NW (Suite 1200), Washington DC 20036
Tel: 202/466-2299/98
Fax: 202/466-8746
CBEMA

TYPEWRITER KEYBOARD CONFIGURATION QUESTIONNAIRE

Name of Organization	Nature of Your Organization (i.e., medical, gov't., legal)	Title of Position

1. Which of the following characters would you most like to have added to the standard typewriter keyboard? Number in order of preference from 1 (most useful) to 11 (least useful).

____ Left bracket [	____ Degree °	____ Greater than >
____ Right bracket]	____ Plus-minus ±	____ Less than <
____ Paragraph ¶	____ Superscript 2 ²	____ Product Dot •
____ Section §	____ Superscript 3 ³	

2. Are there other characters not included in No. 1 which would be more useful? List them below:

3. What characters, if any, do you feel could be eliminated from the existing typewriter keyboard? This should include characters which are not used or which could be created using other characters. *Example:* Upper case . (period) Upper case , (comma) ½ ¼ @.

4. Adjusting to new characters added and/or old characters being removed would be:

☐ Simple ☐ Moderate ☐ Difficult

5. Would you prefer to have additional keys added to provide additional characters? (The addition of four keys is being considered. Figure 2 shows the location of the proposed keys.)

☐ Yes ☐ No

6. Adjusting to new keys added would be:

☐ Simple ☐ Moderate ☐ Difficult

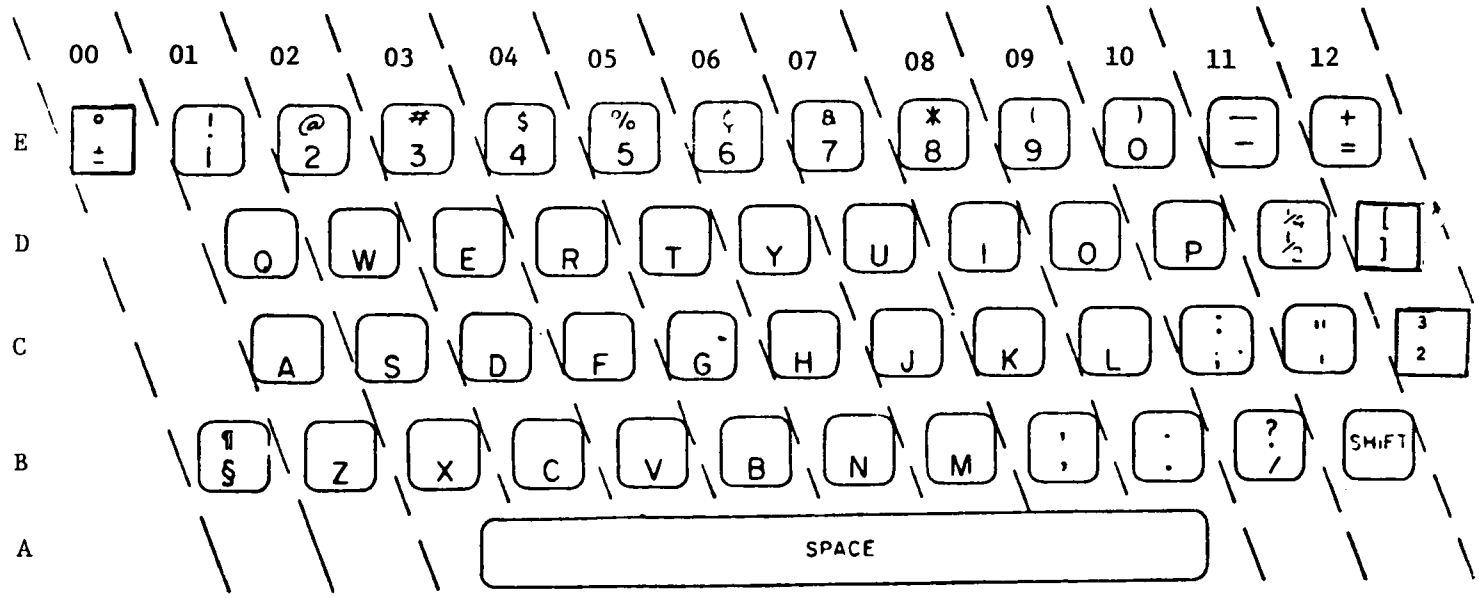


Fig. 2

Proposed Word Processing Keyboard Arrangement

Shape or location of control keys (tab, lock, shift, return) would need to be altered to accomodate the addition of four keys.

STANDARDS ANNOUNCEMENT

A new technical committee of ANSI X3 is being formed to define the character set for optical character recognition. If you would be interested in participating on the committee or on commenting on their "work in progress" please contact:

Patricia M. Caroom
DECUS Standards Coordinator
Box 13047, Capitol Station
Austin, Texas 78711

1. Identification of Proposed Standards Project:

.1 Title: CHARACTER SET (MATRIX PRINTED) FOR OPTICAL CHARACTER RECOGNITION (OCR-M).

.2 Proposer: Technical Committee X3A1 on OCR

.3 Date: 1976-09-29

2. Description of Proposed Standard Project:

.1 Purpose: This project will provide for the development of a stand-alone American National Standard for a matrix printed character set(s). It will provide for a full alphanumeric character set repertoire. The standard will be a graphic shape standard and will not specify any specific matrix organization (size) so as to include the lowest resolution fonts feasible.

.2 Goals: To provide a source of reliable information that will enhance the use of OCR technology in data entry application.

.3 Nature of the Standard: The standard will specify specific font shapes that will be applicable to a majority of matrix printers and will provide for optimum OCR readability.

.4 Level of Activity: The resulting standard, when completed, will be forwarded to ECMA/TC4 as information, and to ISO/TC97/SC5 as a contribution for consideration as an international standard.

3. Benefits:

.1 Intrinsic: The completion and adoption of a matrix standard will provide a set of graphic shapes for use in data entry for OCR systems as provided by matrix printing devices.

.2 Education: The availability of this standard in the matrix printing area should be beneficial to both suppliers and users as a point of reference for equipment compatibility.

3. Benefits (cont'd.):

.3 Economic: The existence of a standard for a matrix printed character set without specifying a particular matrix size should be beneficial to both suppliers and users in the selection of lower cost systems.

.4 Interchange: A standard character set would enhance the usefulness of matrix printing devices in data processing systems. Turn-around forms will be more useful if they can be recaptured directly by OCR readers in the entry step in data processing.

4. Development Feasibility:

.1 State of the Art: Matrix printing developments are presently in an unsettled state of the art due to the rapid appearance of improved devices and processes for segmental printing of characters. Some of these devices and technical procedures are available, and de facto practices may appear.

.2 Resources: Representatives of matrix printing equipment manufacturers have joined the technical committee in addition to OCR reader manufacturers and users and more will be available. Capabilities for the analysis of the various character shapes and character separability are available.

.3 Cost: The cost of development will be nominal and will be primarily borne by the supporting manufacturers of matrix printers and OCR readers. Added resources may be required.

5. Implementation Feasibility:

.1 Supplier Conformance: Requirements for a standard have not yet been developed so that questions of conformance problems are not in evidence. Given continued supplier participation, it would appear that a supplier-acceptable solution would ensue from any development work.

.2 User Operational Considerations: With the lack of specific requirements it is not yet possible to assess user considerations. The participation of users should at least identify potential problem areas.

.3 Legal Considerations: None have appeared to date.

6. Maintenance:

.1 Extent and Frequency of Anticipated Changes: Since work on a standard has not yet begun, it is premature to raise issues relating to its maintenance. In general, it can be foreseen that some revisions may be required to reflect developments of the technology and its application to OCR areas.

.2 Resources: Technical committee membership must retain adequate technical representation after a standard is developed.

.3 Cost: It is not possible to estimate costs at this time.

7. Closely Related Standards Activity:

Continuing liaison with the ECMA/TC4 activities is important. Added liaison with FIPS TG-11 and ISO/TC97/SC3 will be involved.

8. Recommended Time Frame:

It is believed that the development activity will submit a draft proposed standard on a matrix character set (OCR-M) in 1978.

November 9, 1977

Mr. Bob Branton
3115 Brincrest Drive
Dallas, Texas 75234

Dear Mr. Branton:

I am trying to convert an existing tape-oriented Basic-Plus system to use 2780 software and hardware on a PDP-11 570, using version 6B of RSTS/E, to communicate with IBM 370 under JES. Does anyone have technical documentation? Has anyone had experience with this sort of conversion (tape to 2780) and if so, could they write me? Thank you.

Doreen O'Bara

Doreen O'Bara
86 Grace Road
Lake Hiawatha, N.J.
07034

I have just completed a course in Macro-11 under RSTS/E at a local high school with its own RSTS/E system. I did this as project work for Worcester Polytechnic Institute where I am a undergraduate CS major.

The course covered: the PDP-11 instruction set, relevant system operations, user/run-time communications, and run-time systems.

The students periodicly wrote, assembled, linked, debugged, and ran their programs, all under RSTS/E.

I have developed a small run-time system (1K) which provides: diagnosic information, an ODT hook, and a READ and WRITE routine that are callable from the users' programs. These read and write routines allow the students to be able to start programming eariler in the course without the knowledge of XRBs and FIRQBs.

This high school is a real hostile environment as far as system security is concerned. However, there has not been a single incident of system security being broken or crashes suggesting that a user's program was at fault.

I wish to have contact with any persons who have any interests in having general user type programs running under RSTS/E. My address and telephone number is:

Robert Oriol
WPI/Box 1104
Worcester, MA 01609

Zimmerman Metals Inc.
201 East 58th Avenue, Denver, Colo. 80216 / 303-893-3123

December 16, 1977

Murrary Stinson
Newsletter Editor
Scientific Placement, Inc.
2700 S. Post Oak Rd. #2430
Houston, Texas 77056

Dear Murray:

I had a talk with Joan Salkin, Chairman of the RSTS/E Large Users SUB - SIG and I volunteered (was drafted?) to head a 'CUSP' Newsletter. The newsletter would contain hints on changes to systems cusps or suggestions concerning new ones. I would be glad to put out the newsletter from my office here in Denver. I think the area of CUSPS -- old & new was a resounding theme at the RSTS Sig meeting at Fall Decuss '77.

Any interested persons, please contact me at the above address with your comments or ideas for this badly needed newsletter.

Sincerely,

Bob Moore
EDP Manager

BM/ct

Two "hidden" features crept into the RSTS/E V06C-03 release:

1) The "fancy" null job.

On processors with displayable lights (the PDP-11/45, /50, /55, /70) the standard RSTS/E null job shows a left shifting pattern in the "DATA" lights when "DATA PATHS" is selected with the lower rotary switch. This null job costs a total of 5 words.

At a cost of 154(10) words, a much fancier null job can be included in your system.

In addition to the left shifting pattern in the "DATA" lights, the fancy null job displays a right shifting pattern in the "MEMORY ADDRESS" lights. (You must select something like "SUPER I" with the upper rotary switch.) This yields a clockwise turning rotary pattern through both sets of lights. The effect is quite pleasing to the eye if your system runs with any null time.

Furthermore, this "fancy" null job runs in SUPERVISOR mode! Since users run in USER mode, the monitor runs in KERNEL mode, and the null job runs in SUPER mode, it is very easy to tell where your processor is spending its time by glancing at the "mode" lights. (Note: The standard RSTS/E null job runs in KERNEL mode just like the rest of the monitor.) RSTS/E is not only the first DEC operating system to use SUPERVISOR mode, but also the only operating system on PDP-11's to rotate the memory address lights; both are non-trivial feats.

This fancy null job automatically detects the presense of SUPERVISOR mode and reverts to the old style null job if the current processor doesn't have it. Because of this, the prebuilt SYSGEN monitor has been shipped configured with this fancy null job. When you bring up the SYSGEN monitor, set the rotary switches as indicated above. If you like the effect and can waste 149 words, answer the following SYSGEN dialog question as specified:

```
Generate monitor ?      #Y #     Y/L
```

Note also that this fancy null job does not load the system any more than the ordinary null job does (which is no loading at all). Its presence will not degrade system performance.

2) CTRL/T Mini-SYSTAT.

A "hidden" terminal service option costing 412(10) words will add a mini-SYSTAT capability to your RSTS/E system.

If a terminal is connected to a job as the job's console terminal (i.e., the job's KB:), and the new CTRL/R for re-type feature is enabled for that terminal (via TTYSET), and the terminal is not currently in some "funny" mode (e.g., ODT sub-mode, TECO mode, ECHO CONTROL mode, etc.), typing a CTRL/T will produce a "mini-SYSTAT" on the terminal. This is done directly by the terminal service. The only effect it can have on your running program is to change the position of the terminal's "print head" (cursor).

The format of the mini-SYSTAT produced is:

```
jjjjjj+rrrrrr   ss fnn  p(q)K+rK        tt.t(+ii.i) pp
                ss(ccx)
                FP(fff)
```

Where:

jjjjjj is your program's name (e.g., QUE, DIRECT, MACRO, etc.).

rrrrrr is your program's Run-Time System's name (e.g., BASIC, RT11, etc.).

ss is your job's state. The job states are the same as those displayed by the SYSTAT and VTxxPY CUSPs (e.g., RN, KB, ^C, etc.).

fnn is your swap slot number if you are swapped out. Once again, this is the same as SYSTAT reports (e.g, A10, C5, etc.).

ss(ccx) is an alternate job state description if your job is in an I/O wait. The ss is the type of I/O wait (e.g., LP, DB, DK, etc.), the cc is the channel number of the pending I/O operation (e.g., 0, 1, 2, etc.), and the x is either a R or W saying whether the I/O operation is a read or write.

FP(fff) is another alternate job state description if your job is in a FIP (file processor) wait. The fff is the FIP function being performed. The possible functions are:

CLS	Close.
OPN	Open existing file.
CRE	Create a new file.
DLN	Delete a file.
REN	Rename a file.
DIR	Directory lookup.
ERR	Lookup error message.
RST	Close (reset) all channels.
LOK	Lookup a file.
ASS	Assign a device.
DEA	Deassign a device.
DAL	Deassign all devices.

CRT	Create temporary file.
CRB	Create binary (executable) file.
RUN	Open a binary (executable) file for execution.
SND	Send a message.
RCV	Receive a message.
NET	Perform DECNET/E function.
DSP	Everything else... (i.e., most SYS calls)
???	The function was not decodeable.

p(q)K+rK are your program's and Run-Time System's sizes. The p is your program's size in K words, the q is your current maximum program size in K words, and the r is the Run-Time System's size in K words.

tt.t(+ii.i) is your CPU time. The tt.t is your total CPU time to date in seconds (and tenths of a second). The ii.i is your incremental CPU time since the last CTRL/T.

pp is your job priority (e.g., -8, -16, +Ø, +32, etc.).

The 412 word size of the CTRL/T mini-SYSTAT option comes out of the terminal service phase of the monitor. This means that it does not count against the small buffer pool, etc. Almost every monitor will have 412 words "free" in its terminal service phase. The cost is therefore only in total physical monitor size. Note that the total physical monitor size may not increase with the addition of this feature. The RSTS/E monitor consists of four (or more) phases. Each phase is rounded up to be a multiple of 256(1Ø) words in size. The sum of all of the phases is rounded up to be a multiple of 1Ø24(1Ø) words. Therefore, the addition of 412 words to the terminal service phase will make that phase either 256 or 512 words larger. If that increase does not make the sum of all phases 1Ø24 words larger, you can include the mini-SYSTAT feature with no penalty at all.

Unlike the "fancy" null job, this CTRL/T mini-SYSTAT option is not configured in the SYSGEN monitor.

To configure the CTRL/T mini-SYSTAT option in your monitor, answer the SYSGEN dialog question as specified:

```
Generate monitor ?      #Y #     Y/T
```

or:

```
Generate monitor ?      #Y #     Y/L/T
```

to include both the "fancy" null job and the CTRL/T mini-SYSTAT.

Note: There are two small, non-fatal bugs in the CTRL/T mini-SYSTAT option. The totally non-supported patch that follows corrects these bugs.

CTRL/T MINI-SYSTAT ERRORS - OPTIONAL TERMINAL SERVICE PATCH

PROBLEM:

The optional CTRL/T mini-SYSTAT feature of the terminal service incorrectly reports the read versus write status of the pending I/O when a job is in an I/O wait.

In addition, the column alignment is incorrect after printing the pending FIP function when a job is in FP wait.

SOLUTION:

This optional terminal service patch corrects these problems.

PROCEDURE:

1. This is an optional patch to the RSTS/E V06C-03 executive. It may be installed in any target monitor SIL configured with the CTRL/T mini-SYSTAT feature.

2. Bootstrap your system disk and use the PATCH option as shown below.

```
Option: PATCH
File to patch? <lf>      (LINE FEED for installed monitor SIL)
Module name? TER
Base address? TTSYST
Offset address? 526
 Base   Offset  Old     New?
??????  000526  000020  ? 16
??????  000530  000001  ? ^Z            (CTRL/Z for new offset)
Offset address? 670
 Base   Offset  Old     New?
??????  000670  004567  ? 112602
??????  000672  000504  ? 1417
??????  000674  000040  ? 10246
??????  000676  112602  ? 4737
??????  000700  001414  ? TTDVR+177770
??????  000702  010246  ? 111602
??????  000704  142716  ? ^Z            (CTRL/Z for new offset)
Offset address? ^Z                     (CTRL/Z for new base)
Base address? TTDVR
Offset address? 177770
 Base   Offset  Old     New?
??????  177770  000000  ? 4537
??????  177772  000000  ? TTSYST+1400
??????  177774  000000  ? 40
??????  177776  000000  ? 207
??????  000000  ??????  ? ^C            (CTRL/C to exit)

Option:
```

digital

1 JAN 78

Murray Stinson
Scientific Placement, Inc.
2700 S. Post Oak Road, #2430
Houston, TX 77056

Dear Murray,

During my talk at the San Diego DECUS symposium, I mentioned a program that converts RUNOFF documents to BASIC-PLUS comments. A few people there thought the program was interesting, and I am enclosing a listing that you may publish in the Newsletter for their benefit.

Please note that the program was written very quickly, and was tested even more quickly.

Last, but not least, may I extend my thanks to all the "RSTS people" for a wonderful time in San Diego.

Sincerely yours,

MARTIN MINOW

DIGITAL EQUIPMENT CORPORATION, 146 MAIN STREET, MAYNARD, MASSACHUSETTS 01754
(617)897-5111 TWX: 710-347-0212 TELEX: 94-8457

```
1       !
        !       C O M E N T   C O N V E R T E R
2!              PROGRAM          : COMENT
5!              VERSION          : V06C
6!              EDIT             : 03
7!              EDIT DATE        : 15-NOV-77
8!
9!              AUTHOR           : MAM
10      EXTEND
100     !
        !       I N S T R U C T I O N S
111!
   !     The COMENT program converts RUNOFF output to comment lines
   !     for inclusion in source programs. The RUNOFF source file
   !     (.RNO) should have the following standard specifications:
   !
   !             .NNM.PS 55,69.LM 8.RM 69
   !
   !     Note that the first eight (blank) columns will be stripped
   !     off by this program. Left and right margin commands should
   !     be relative, rather than absolute; the right-margin being
   !     three less than the "real" right margin. The above example
   !     is satisfactory for VT05's. Also, you should avoid fancy
   !     stuff like underline.
   !
   !     After creating the .DOC file, run COMENT as follows:
   !
   !             run coment
   !             COMENT -- Input file (.DOC)?  foobar
   !
   !     Enter the .DOC file name, FOOBAR.DOC in the example.
   !
   !             Output file (.TMP)?  foobar
   !
   !     Enter the output file name, FOOBAR.TMP in the example.  Note,
   !     a file name must be given.
   !
   !             First line number <111>?  <CR>
   !
   !     Enter the first line number to be generated or type
   !     carriage-return to accept the default.
   !
   !             Line width for & trailer, zero for <LF>?  <CR>
   !
   !     Type the line width if you want BASIC-PLUS 2-style line ter-
   !     mination.  Type zero (or return) if you want
   !     BASIC-PLUS 1-style line termination.
   !
   !     The program then converts the .DOC file to a BASIC-PLUS com-
   !     ment file (including an END statement at line 32767). To in-
   !     corporate this in your program, OLD the program, APPEND the
   !     comment file, and REPLACE the program source file.
   !
```

```
1000    !
        !       I N I T I A L I Z A T I O N
1010    PRINT "COMENT -- Input file (.DOC)";
        \ INPUT LINE INFILE$\ INFILE$ = CVT$$(INFILE$,-1%)
        \ INFILE$ = INFILE$ + ".DOC"
                        IF INSTR(1%,INFILE$,".") = 0%
        \ OPEN INFILE$ FOR INPUT AS FILE 1%
                ! GET AND OPEN INPUT FILE
1020    PRINT "Output file (.TMP)";
        \ INPUT LINE OUTFILE$\ OUTFILE$ = CVT$$(OUTFILE$, -1%)
        \ OUTFILE$ = OUTFILE$ + ".TMP"
                        IF INSTR(1%, OUTFILE$, ".") = 0%
        \ OPEN OUTFILE$ FOR OUTPUT AS FILE 2%
                ! GET AND OPEN OUTPUT FILE
1030    INPUT "First line number <111>"; FIRST%
        \ FIRST% = 111% IF FIRST% = 0%
        \ INPUT "Line width for & trailer, zero for <LF>"; LWIDTH%
                ! GET THE OTHER PARAMETERS
1100    ON ERROR GOTO 9000
        \ NUMBER% = FIRST% - 1%
        \ BLANK%, TOF% = 0%
        \ TABBER$ = CHR$(9%)
        \ LF$ = CHR$(10%)+CHR$(13%)+CHR$(0%)
        \ FF$ = CHR$(12%)
        \ FF% = ASCII(FF$)
                ! INITIALIZE STUFF
                ! NUMBER%        LINE NUMBER
                ! BLANK%         BLANK LINES COUNTER
                ! TOF%           TOP OF FORM FLAG (NON-ZERO AFTER <FF>)
                ! TABBER$        TAB CHARACTER
                ! LF$            LINE-FEED STRING
                ! FF$            FORM-FEED STRING
                ! FF%            FORM-FEED VALUE
```

```
2000    !
        !       D O   I T

2010    INPUT LINE #1%, DATUM$
        \ TOF% = 1% IF ASCII(RIGHT(DATUM$,LEN(DATUM$))) = FF%
        \ DATUM$ = RIGHT(CVT$$(DATUM$,132%),8%)
        \ IF DATUM$ = "" THEN
          BLANK% = BLANK% + 1%
          \ GOTO 2000
                ! GET INPUT LINE, SET NEW PAGE FLAG IF <FF> SEEN
                ! STRIP OFF TRAILING <CR><LF> AND SPACES
                ! COUNT BLANK LINES (DON'T PRINT THEM YET)

2020    IF TOF% THEN
          PRINT #2%, BLAH$; FNLF$; FF$; IF NUMBER% >= FIRST%
          \ NUMBER% = NUMBER% + 1%
          \ BLAH$ = SPACE$(LEN(NUM1$(NUMBER%))) + "!"
          \ TOF%,BLANK% = 0%
          \ PRINT #2%, NUM1$(NUMBER%);"!";FNLF$;
                ! NEW PAGE SIGNALED...
                ! FINISH OFF LAST PAGE WITH <FF> (UNLESS THIS
                ! IS THE FIRST PAGE).  STEP LINE NUMBER, GET
                ! WIDTH FOR PRETTY COMMENTING, AND PRINT THE LINE.
                ! THEN, CLEAR PAGE FLAG AND BLANK LINE COUNTER.

2030    PRINT #2%, BLAH$; FNLF$; FOR Q% = 1% TO BLANK%
        \ BLANK% = 0%
        \ PRINT #2%, BLAH$; TABBER$; DATUM$; FNLF$;
                ! NON-BLANK LINE, PRINT ANY STORED-UP BLANKS,
                ! THEN PRINT THE COMMENTARY ITSELF


2040    GOTO 2000
                ! GO FOR ANOTHER

9000    IF ERR=11% THEN RESUME 9010
          ELSE ON ERROR GOTO 0
                ! ACCEPT ONLY END-OF-FILE

9010    ON ERROR GOTO 0
        \ PRINT #2%; BLAH$; FNLF$; FF$
        \ PRINT #2%, "32767     END"
        \ CLOSE 1%, 2%


15100   DEF FNLF$
        \ FNLF$ = ""
        \ IF LWIDTH% <> 0% THEN PRINT #2%, TAB(LWIDTH%-2%) " &"
            ELSE PRINT #2%, LF$;
                ! FNLF$ PRINTS THE END OF A CONTINUED LINE,
                ! EITHER <LF> OR AMPERSAND <CR>.

15110   FNEND

32767   END
```

```
#      0       9000   9010
#   2000       2010   2040
#   9000       1100
#   9010       9000
BLAH$          2020   2020@  2030   2030   9010
BLANK%         1100@  2010   2010@  2020@  2030   2030@
DATUM$         2010@  2010   2010   2010   2010@  2010   2030
ERR            9000
FF$            1100@  1100   2020   9010
FF%            1100@  2010
FIRST%         1030@  1030   1030@  1100   2020
FNLF$          2020   2020   2030   2030   9010   15100# 15100@
INFILE$        1010@  1010   1010@  1010   1010   1010@  1010
LF$            1100@ 15100
LWIDTH%        1030@ 15100  15100
NUMBER%        1100@  2020   2020   2020@  2020   2020
OUTFILE$       1020@  1020   1020@  1020   1020   1020@  1020
Q%             2030@
TABBER$        1100@  2030
TOF%           1100@  2010@  2020   2020@
```

Please check that the following variables have been referenced properly:
Q%

```
   16 Variables     75 References    26 Statements
  188 Name bytes   312 Code bytes
   20 Total time     6 CPU time

K-Words Reserved      Used      Free
  Data      2.09      1.43       .66
  Code       .91       .48       .44
 Total      3.00      1.91      1.09
```

```
8 !     Patches to LOGIN by Data Processing Design, Inc.
        Edit #3, 16-Jun-77 TM

        This file when appended to the distribution version
        of LOGIN (DEC edit number 6B-03, though October, 1977 dispatch)
        provides the following facilities:

        1. The 1,2 password permits login to any account
        2. Locked accounts are implemented by the ?????? password
        3. System Not Available is implemented by a SNA??? [1,2] password
        4. Project limits are enforced
        5. A MAIL.TXT file in the user's account is printed if it exists
        6. The program INITIAL is chained to in the user's account if it exists
        7. A Message AUDIT TRAILING the login is sent to a detached receiver

        FEATURE         LINES USED                      VARIABLES USED

          1             8010-8015                       I%, T%, Z1$, J$
          2             8015
          3             8020
          4             8600-8699, 8050-8110            I1%, I2%, I3%, I4%, C%
          5             14000-14900                     V%
          6             9010-9015, 32415
          7             8001-8007, 32417

        NOTE: Audit trailing of logins

                The method used here requires a detached job to be
                a receiver of all AUDIT messages.  Since more of our
                software than LOGIN uses AUDIT, the overhead to use
                it for LOGIN, and also LOGOUT is minor.  The receiver
                program (AUDIT) is not very complicated and it can be
                easily developed.
9 !
        COPYRIGHT (C) 1977   Data Processing Design, Inc.

        Although this program has been tested, no warranty,
        expressed or implied, is made by Data Processing Design as to
        the accuracy and functioning of the program and all
        related program material, nor shall the fact of the
        distribution constitute any such warranty.  This program may
        be copied and distributed without restriction provided that
        the copyright notice is included.

2000    GOSUB 12000
        :GOSUB 8000
        :GOTO 1000 IF I%
        : GOSUB 15000
        : IF I% THEN 900 ELSE
                GOSUB 17100
                : GOTO 5010
                ! Hooked into DPD access check (8000).


5210    I$=SYS(CHR$(0%))+SYS(CHR$(2%))
```

```
        \ CLOSE 1%,2%
        \ PRINT
        \ GOSUB 27100
        \ GOSUB 9000 UNLESS N1%
        \ PRINT CHR$(10%); FOR I%=1% TO 3%
        \ GOTO 32650 UNLESS N1%

        ! The routine at 27000 does the AUDIT trailing of the LOGIN
          while the routine at 9000 does the chain to a program after login

7022 J5%=SWAP%(CVT$%(MID(SYS(CHR$(6%)+CHR$(-3%)),11%,2%)))      ! JOBTBL.
7024 J4%=PEEK(J5%+(J%*2%))                                      ! Address of JDB

7026 IF J4%=0% THEN M%(5%),M%(6%)=0%
        ELSE J4%=PEEK(PEEK(J4%+8%)+24%)
        \       M%(5%)=SWAP%(J4%) AND 255%
        \       M%(6%)=J4% AND 255%


                ! GET job # of job being attached to.

7028 A$=NUM$(M%(5%))+',+NUM$(M%(6%))                    ! Create [P,P]
7030    GOSUB 12000
        \ Z%=-1%\ GOSUB 8000
        \ IF I% THEN PRINT' Failure to attach to job'; J%
        \ GOTO 7000


                ! Go through DPD access checker for attach.

7040    CHANGE SYS(CHR$(6%)+CHR$(-10%)+P$) TO M%
        \ IF M%(28%) AND 154% THEN 7900 ELSE
                M%(5%)=A0%
                \ M%(6%)=SWAP%(A0%)
8000    !*****

        SECTION FOR DPD MODIFICATIONS

        This section through 8699 incorporates the majority
        of modifications.

        ******

8001 Z1$="LOGIN"                        ! Name of program for message send
8002 IF INSTR(1%,P$,'?')<>0% OR INSTR(1%,P$,'*')<>0% THEN 8500

                ! Make sure typed password has no ?'s.

8005 CHANGE SYS(CHR$(6%)+CHR$(-10%)+"["+A$+"]"+P$) TO M%

        ! Convert the account with filename string scan

8006 I$=SYS(CHR$(6%)+CHR$(14%)+STRING$(4%,0%)+CHR$(M%(5%))+CHR$(M%(6%)))

                ! Look up user's password.
```

```
8007 J$=SYS(CHR$(6%)+CHR$(14%)+STRING$(4%,0%)+CHR$(2%)+CHR$(1%))

                ! Look up [1,2] password.

8010 I%=-1%                         ! Start of with flag indicating login failed
8011 GOTO 8015 IF M%(T%+7%)<>ASCII(RIGHT(J$,T%+9%)) FOR T%=0% TO 3%

        ! See if they know the 1,2 password

8012 Z1$=Z1$+"$ "
        \ J$="1,2"
        \ P$=RAD$(SWAP%(CVT$%(RIGHT(I$,9%))))+
             RAD$(SWAP%(CVT$%(RIGHT(I$,11%))))
        \ I%=0%
        \ GOTO 8550

                ! They do-- substitute the user's actual password and do it.

8015 IF "???"=RAD$(SWAP%(CVT$%(RIGHT(I$,9%)))) THEN
        PRINT "?Account locked."
        \ GOTO 32650

        ! A password with ??? indicates a locked account.

8020 IF M%(6%)>1% AND "SNA"=RAD$(SWAP%(CVT$%(RIGHT(J$,9%)))) THEN
        PRINT "?System not available."
        \ GOTO 32650

        ! If the first three characters of the 1,2 password are
          SNA and the user did not know the 1,2 password, tell
          them that the system is not available, and don't let them
          log in.

8025 I%=0%                          ! Set the flag to indicate login success
8035 GOTO 8500 IF M%(T%+7%)<>ASCII(RIGHT(I$,T%+9%)) FOR T%=0% TO 3%

        ! Did they type their own password?

8040    IF ASCII(MID(SYS(CHR$(6%)+CHR$(9%)),2%,1%))=0% OR Z%
                THEN 8550

                ! If on KB0 or attaching, dont check project limits.
                  NOTE: This could be modified to not check on other
                        keyboards as well.

8050 RESTORE
        \ I1%=-1%

        ! This is the start of the project limit checking section

8060 READ I1%,I2%
                UNTIL I1%=M%(6%) OR I1%=0%
        \ IF I1%=0% THEN 8550
```

```
                ! Read a project and limit.  If no more, no limit enforced.

8070    C%=0%
        \ I3%=SWAP%(CVT$%(MID(SYS(CHR$(6%)+CHR$(-3%)),11%,2%)))

                ! Get ready to see how many other jobs are logged into that prj.

8080    FOR I4%=I3% STEP 2% UNTIL PEEK(I4%)=-1%
                \ I3%=PEEK(I4%)
                \ IF I3%
                        THEN IF I1%=(SWAP%(PEEK(PEEK(I3%+8%)+24%)) AND 255%)
                                THEN C%=C%+1%

                ! Scan down the JOBTBL and count the jobs in the project.

8090    NEXT I4%
8100 IF C%>=I2%
        '       THEN PRINT '?Too many jobs already on this project.'
                \ I%=-1%

        ! If the limit has been reached, say so, and set failure flag.

8110 GOTO 8550
8500 I%=-1%
        \ IF NOT Z% THEN
          PRINT "?Invalid entry."
        \ Z1$=" LF: "+NUM1$(M%(6%))+","+NUM1$(M%(5%))+"/"+CVT$$(P$,-1%)+"/ "
        \ GOSUB 27100

                ! We get here if the login failed for some reason, in this
                  case we still audit trail the login by setting Z1$ to
                  LF (Login Failed) indicating the attempted account and
                  the password that was typed.

8550    Z%=0%
        \ RETURN
8600 !
        The following data statements define the projects which
        have a limit on the number of jobs permitted.
        Entries have the form: <proj>,<limit>


8610    DATA    10,4
8620    DATA    20,1
8630    DATA    30,1
8698 ! All project limit data statements must come before this line.
8699    DATA    0,0
9000 !
        This section is entered after the job is logged in, if the
        user did not use "/" to login.  It trys to chain the program
        "INTIAL" in the user's account.  Since at this point we know
        what terminal and account the user is running under, this code
        could be given "smarts" to select certain program names for
        certain terminals or accounts.
```

```
9015 CHAIN "INTIAL"

                        ! Try to chain to the user's program.

9055 RETURN             ! Here if the chain failed.
13999 !
        This section not only prints $NOTICE.TXT, but prints
        the file MAIL.TXT in the user's accounts if it exists.
        Our mail system stores the user's messages in such a
        file in their account.  This modification could also be
        used to print a project level notice files such as #NOTICE.TXT.

14000   IF N% THEN
                OPEN "$NOTICE.TXT" FOR INPUT AS FILE 2%: V%=0%
14010   INPUT LINE #2%, I$
        : PRINT I$;
        : GOTO 14010
14800 V%=-1%
        \ OPEN 'MAIL.TXT' FOR INPUT AS FILE 2% IF N%
        \ GOTO 14010

        ! The V% flag says we're reading MAIL.TXT rather than NOTICE.TXT$

14900   CLOSE 2%
        \ IF V% THEN RESUME 19999
                ELSE RESUME 14800

        ! If the V% flag is set, there was no MAIL.TXT file.

15000   I%=-1%
        \ CHANGE SYS(CHR$(6%)+CHR$(-10%)+"["+A$+"]"+P$) TO M%
        \ RETURN IF (M%(28%) AND 26%)<>0%
        \ M%(1%)=6%
        \ M%(2%)=4%
        \ CHANGE M% TO I$
        \ CHANGE SYS(I$) TO M%
        \ A%=PEEK(PEEK(PEEK(520%)+8%)+24%)
        \ I%=0%
15010   J%=16%
        \ J%=28% IF (A% AND -256%)=256%
        \ I$=SYS(CHR$(6%)+CHR$(-13%)+CHR$(-1%)+
                CHR$(-1%)+CHR$(-8%)+
                CHR$(-1%)+CHR$( 6%)+
                CHR$(-1%)+CHR$( J%))
        \ RETURN

        ! Set the user's priority, run burst, and swap max.
          Note that the swap max is 28 if the user is in [1,*]
          and 16 otherwise.

27100 !
        This is the subroutine that does the AUDIT trailing for
        LOGINS (both successful and failures).  The auditing is
        done by sending a message to a detached receiver named "AUDIT"
```

```
        which is started in a privileged account when the system comes up.

        Note that if AUDIT is not running, no error message is generated
        and the login still succeeds.  The message is of the form:

        Header:  ILOGIN
                 CHR$(13)
        Acct:    proj,prog
                 CHR$(13)
        KB:      kb number
                 CHR$(13)
        message: (depends on LOGIN success or failure; = Z1$)
        Misc:    TIME$(0)
                 CHR$(13)
                 "INOUT" (filename used for AUDITing)

27102   ON ERROR GOTO 27110
        \ Z%=PEEK(PEEK(PEEK(520%)+8%)+24%)
27104   Z$="I"+"LOGIN "+CHR$(13%)+NUM1$(SWAP%(Z%) AND 255%)+
           ","+NUM1$(Z% AND 255%)+CHR$(13%)+
           NUM1$(ASCII(RIGHT(SYS(CHR$(6%)+CHR$(9%)),2%))/2%)+
           CHR$(13%)+Z1$+TIME$(0)+CHR$(13%)+"$INOUT"

        ! Build the message string, it should be less than 64 characters

27106   Z$=SYS(CHR$(6%)+CHR$(22%)+CHR$(-1%)+CHR$(0%)+"AUDIT "+
                 STRING$(10%,0%)+Z$)

        ! Send the message, errors are trapped at line 27110

27108 ON ERROR GOTO 32400
        \ RETURN
27110 RESUME 27108

        ! Here if we get an error sending the message.  Another
          option is to send a message to KB0: so the operator of
          system manager will know that the AUDIT feature is not working.

32015   I1%=-1%
        \ READ I1%,I2% WHILE I1%<>0%     ! Read past proj. limits.
32415 IF ERL=9015% THEN RESUME 9055

        ! Can't chain to INTIAL.

32417 IF ERL=8006% THEN RESUME 8040

        ! No such account on password lookup. Let the LOGIN SYS() error.
```

A BYTE OF INSIGHT

This article begins a series of in depth technical papers by Mike Mayfield and Rich Marino of Data Processing Design, Inc., Buena Park, CA. These papers are intended to help provide information for the advanced RSTS analyst or software manager.

Future articles will include information on some of the following topics:

System Performance Analysis
Program Optimization
Macro Under RSTS
Internal Monitor Tables
Monitor EMT Calls
Writing a Runtime System
Writing a Device Driver
Advanced Programming Techniques
Care and Feeding of Programmers
System Management
Memory Management Internals
Scheduler Internals

As you can see from the list above there's a lot to cover. Considering the number of times the newletter is released per year it could take several years to complete this list and, therefore, it probably won't be finished. I would greatly appreciate your input on those topics you would like to see covered or on any changes you'd like to see made.

Please send any comments to:

Mike Mayfield
Data Processing Design, Inc.
6980 Aragon Crl., Suite 8
Buena Park, CA 90620

Although every effort has been made to ensure that the information given in these articles is accurate, there will undoubtedly be some mistakes or misinformation. Please excuse these, as we are many times using information which can be gathered only by much experimentation and just plain guessing. I think you'll find we're right most of the time.

If anyone has information to add to that which we're giving here, please send it to the address above and we'll try to include it, where appropriate, in future articles.

Disk Internals

Almost every system I've seen that has reached capacity has done so, not because of lack of CPU capability, but because of excess disk I/O. Optimal use of the disk can make or break a system. For this reason I have decided to start this series by discussing the internal operation of the file system under RSTS.

The RSTS file system is based on a master file directory (MFD), user file directory (UFD) hierarchy. The MFD is used to store information about each of the accounts. It contains all of the accounting information for each account, as well as a pointer to where the UFD is physically located on the disk.

The UFD contains information about each file in a user's account. Information such as creation date, file size, and the physical location of the file on the disk are stored in the UFD.

Figure 1 shows a graphical representation of this directory structure. The parts of the directory will be discussed in detail later.

Before we continue, a few terms are needed:

Cluster: A number of contiguous blocks (See the RSTS Manager's Guide).

Device Cluster: The minimum size of a cluster for a particular device. A device cluster is the smallest unit of disk space directly addressable using 16 bits.

Device Cluster Number (DCN): Device clusters are numbered sequentially starting at zero.

Blockette: The smallest unit of information used in an MFD or UFD. Each block of the MFD or UFD is divided into 32 blockettes of 16 bytes each.

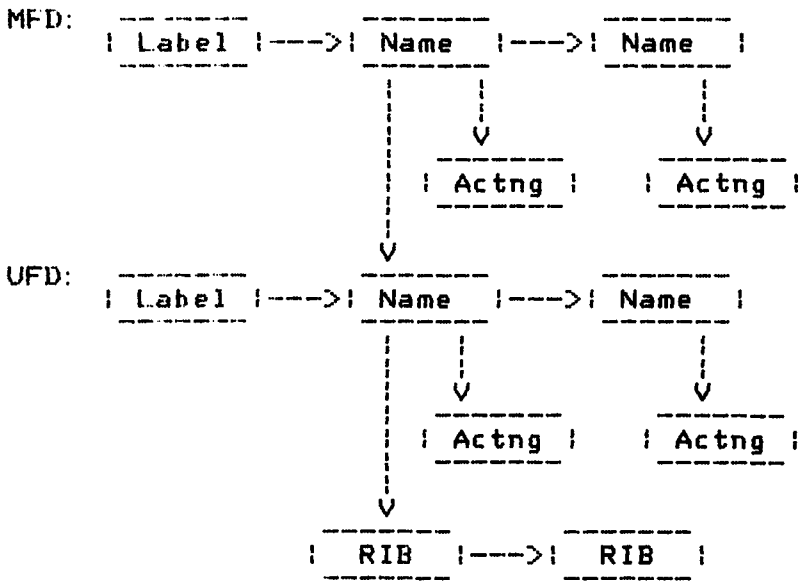


FIGURE 1: Graphic Representation of MFD and UFD

Master File Directory (MFD)

The master file directory is used to store information about each account on the system. It is also used to locate the file information for each account.

The MFD is the only portion of RSTS, except for the bootstrap, that must reside at a fixed location on the disk. The first cluster of the MFD is always located at DCN 1. The remaining clusters are located by following links in the MFD structure and may be located anywhere on the disk.

The MFD consists of up to six different types of blockettes: the Label Block, Name Block (NB), Accounting Block (AB), Attributes Block (AT), Cluster Map Block (CM), and Unused blocks. In addition the MFD may also contain file information for the [1,1] UFD.

MFD LABEL BLOCK

The MFD label block contains the information needed to mount the disk pack. It is always the first blockette in MFD cluster 0.

1	Link to First NB in MFD	0
3	-1	2
5	0	4
7	0	6
11	Pack Cluster Size	10
13	Pack Status	12
15	Pack ID (First word, in RAD50)	14
17	Pack ID (Second word, in RAD50)	16

Link to First NB in MFD:

Link to the first name block in the MFD. This link will never be zero since there is always a name block for account [1,1].

Pack Cluster Size:

The minimum cluster size for files created on this pack. The PCS must be at least the device cluster size.

Pack Status:

The pack status is a bit pattern as follows:

<15> Bit 15 is set to 1 on a mount and 0 on a dismount. If an attempt is made to mount a pack that was not properly dismounted this bit will show that a "clean" is needed.

<14> Bit 14 is set to 1 on all private disks and cleared on all public disks. The system disk has bit 14 set as if it were a private disk, thus allowing it to be mounted while another system disk is in use.

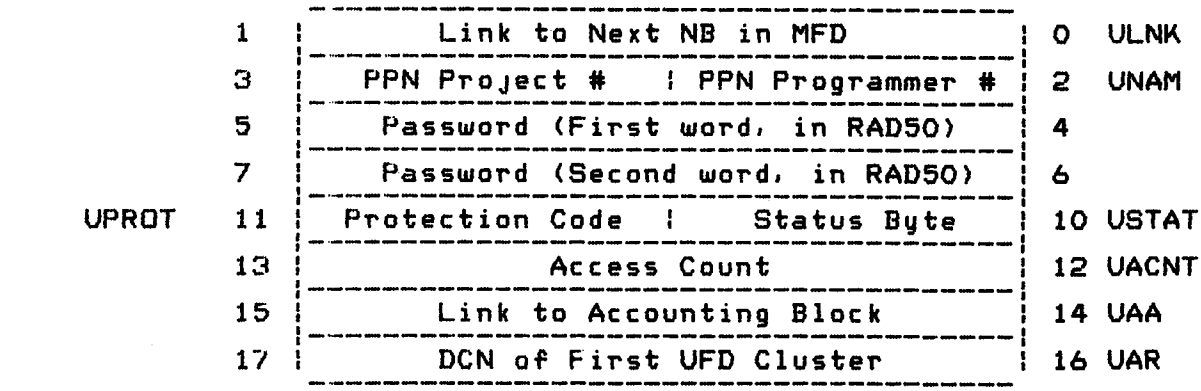
<13:0> Not used (Must be zero)

Pack ID:

The logical pack ID is stored as two words of RAD50. The pack ID is used when the disk is mounted and as an alternate name for the pack once it has been mounted.

MFD NAME BLOCK

The MFD Name Block is used to catalog all the accounts in the system. Each account has a name block associated with it. The name blocks are linked together in a list in the order they were created. The name block contains all the information necessary to identify a desired account.



Link to Next NB in MFD:

Each name block in the MFD is linked together by this word (see Directory Link described later). The last link will be zero.

PPN Project and Programmer #:

The account PPN is stored as two seperate bytes. A PPN of [0,0] is illegal, as well as [255,*] and [*,255].

Password:

The password is stored as two words of RAD50. A password of ?????? is legal, but LOGIN treats this as a "locked" account.

Status Byte:

The status of files and UFD entries are stored as a group of bits as follows:

<0> File data space is on another disk.

MFD: Always 0
UFD: Files such as SWAPn.SYS, OVR.SYS, ERR.SYS, and BUFF.SYS may be on a swapping disk. Files must be in [0,1] to be on a swapping disk.

<1> Not used (Must be zero)

<2> File (or UFD) open for write access

<3> File (or UFD) open for update

<4> File may not be extended

MFD: Always 1
UFD: A 1 specifies that the file is contiguous

<5> File is non-deletable

MFD: Always 1
UFD: This bit is set and cleared by the REFRESH Init option

<6> Name block for UFD. This bit specifies that the NB is for a UFD. If it is off, then the NB is for a file rather than a UFD.

MFD: Always 1
UFD: Always 0

<7> File is marked for deletion. Files are not actually removed from the UFD until their access count goes to 0.

Protection Code:

The UFD's protection code as one byte. UFD's may have a protection code, but it is never used since you must be privileged to open a UFD. The execute bit <64> and the privileged bit <128> are always 0 for a UFD name block.

Access Count:

The access count specifies the number of users currently logged in under this account.

Link to Accounting Block:

Link to the Accounting Block for this account.

DCN of First UFD Cluster:

This word contains the device cluster number for the first cluster of the UFD. If no files have ever been created in this UFD since it was created or "zeroed" this word will be zero. Space for the UFD is not actually allocated until its first file is created.

MFD ACCOUNTING BLOCK:

The MFD Accounting Block contains usage accounting information about each account. This is the information used by MONEY to report account usage.

1	Link to AT for UFD !0!0!0!1	0	ULNK
3	Accumulated CPU Time (LSB)	2	MCPU
5	Accumulated Connect Time	4	MCON
7	Accumulated Kilo-Core-Ticks (LSB)	6	MKCT
11	Accumulated Device Time	10	MDVT
13	CPU Time (MSB) ! KCT (MSB)	12	MMSB
15	Logout Disk Quota	14	MDPER
17	UFD Cluster Size	16	UCLUS

Link to Attributes Block:

Although UFD's normally don't have attributes, it is theoretically possible to add them. This allows additional information to be kept with each UFD (by a user written program).

Accumulated CPU Time (LSB):

This word combines with the MSB byte below to form a three byte value for accumulated CPU time. This number can be in the range 0-16,777,215 1/10 seconds.

Accumulated Connect Time:

This word contains the amount of connect time, in minutes, used by this account.

Accumulated Kilo-Core-Ticks:

This word combines with the MSB byte below to form a three byte count of kilo-core-ticks. This number can be in the range 0-16,777,215.

Accumulated Device Time:

This word contains the amount of device assignment time, in minutes, used by this account.

Logout Disk Quota:

This word specifies the number of blocks the account can retain when a user logs out. The monitor does not enforce this restriction. The LOGOUT cusp checks this word before allowing a user to logout.

UFD Cluster Size:

This word contains the cluster size for this UFD. When the monitor expands the size of the UFD it uses this value to determine the number of blocks to allocate. An improper UFD cluster size can have a significant impact on disk overhead.

MFD CLUSTER MAP BLOCK:

The Cluster Map is used to locate the clusters of an MFD. There is a cluster map block at the end of every block of the MFD, starting at offset 760(8).

1	MFD Cluster Size	0
3	DCN of MFD Cluster 0	2
5	DCN of MFD Cluster 1	4
7	DCN of MFD Cluster 2	6
11	DCN of MFD Cluster 3	10
13	DCN of MFD Cluster 4	12
15	DCN of MFD Cluster 5	14
17	DCN of MFD Cluster 6	16

MFD Cluster Size:

This word contains the cluster size for the MFD. It is used by FIP any time the MFD is extended.

DCN of MFD Cluster n:

The Device Cluster Number of the appropriate cluster is contained a corresponding entry of the last 7 words of the CM. Use of these pointers allows access to any block of the MFD in at most one seek.

UNUSED BLOCKETTES

Blockettes which have never been used, or have been returned to the free list by a file deletion, are marked as available for use. When a new blockette is needed, the current block is searched for an Unused Blockette. If it isn't found in the current block, the remainder of the MFD is searched. If no unused blocks are found in the entire MFD, it is extended, if possible.

1	0	0
3	0	2
5	Unused (Contents are undefined)	4
7	Unused (Contents are undefined)	6
11	Unused (Contents are undefined)	10
13	Unused (Contents are undefined)	12
15	Unused (Contents are undefined)	14
17	Unused (Contents are undefined)	16

User File Directory (UFD)

The UFD is used to catalog a user's files. Each file in an account has an entry in the UFD which completely describes that file.

Improper arrangement of files in the UFD can have a dramatic effect on the efficiency of disk I/O. If properly done, disk I/O requires no overhead at all. But improperly done (alas, the obvious way) you can have dozens of overhead seeks for every "GET" or "PUT" in your program.

File information in the UFD is kept in a linked list. In most cases, new files are added to the end of the list, although it is possible with an optional monitor patch or an optional mode to add files at the head of the list instead.

The UFD is almost identical in format to the MFD. It contains a Label Block, Name Blocks (NB), Accounting Blocks (AB), Attribute Blocks (AT), Cluster Maps, and Unused Blocks. In addition, it contains Retrieval Blocks (RIB) which are used for determining the actual disk location of blocks in the data files.

UFD LABEL BLOCK

The first blockette in cluster 0 of each UFD contains the root link to the first name block in the UFD. No other information is currently used in the label block.

1	Link to First UFD Name Block	0
3	-1	2
5	0	4
7	0	6
11	0	10
13	0	12
15	0	14
17	0	16

Link to First UFD Name Block:

This word links to the first Name Block in the UFD.

UFD NAME BLOCK

Every file in the UFD has a Name Block associated with it. This blockette contains all the information necessary to identify the file it belongs to. When opening a file, FIP follows the linked list of name blocks searching for the desired file.

1	Link to Next UFD Name Block	0	ULNK
3	File Name (First Word, RAD50)	2	UNAM
5	File Name (Second Word, RAD50)	4	
7	Filename Extension (RAD50)	6	
11	Protection Code ! Status Byte	10	USTAT
13	File Access Count	12	UACNT
15	Link to Accounting Block	14	UAA
17	Link to First RIB	16	UAR

Link to Next UFD Name Block:

This word contains a pointer to the next name block in the UFD. If this is the last file in the UFD the link will be zero.

File Name and Extension:

The filename and extension are stored as three words of RAD50 characters. Filenames containing "invalid" characters such as "?" and "." are possible but BASIC+ checks for them using the filename string scan monitor call.

Status Byte:

The current file status is stored as a byte. See the description of this byte in the MFD Name Block.

Protection Code:

The file's protection code is stored as a byte.

File Access Count:

This word denotes the number of times this file is currently open. If, on a file open, the access count is nonzero the status byte is checked to see if the file is open in write or update mode. If the file is marked for deletion, it will actually be deleted when the access count goes to zero on the last close.

Link to Accounting Block:

This word links to the Accounting Block for this file.

Link to First RIB:

This word links to the first Retrieval Information Block for this file.

UFD ACCOUNTING BLOCK

Every file in the UFD has associated with it an accounting block. This block contains all the "usage" type information about the file.

1	Link to Attributes Block !0!0!0!1	0	ULNK
3	Last Access Date	2	UDLA
5	Number of Blocks in the File	4	USIZ
7	Creation Date	6	UDC
11	Creation Time	10	UTC
13	Runtime System Name (First Word)	12	URTS
15	Runtime System Name (Second Word)	14	
17	File Cluster Size	16	UCLUS

Link to Attributes Block:

This word contains a link to the first attributes block if the file has attributes. If not the link will be zero.

Last Access Date:

This word is set to the current system date whenever a file is accessed. If the "Last Modified Date" option is used, then this word contains the date the file was last modified, rather than accessed.

Number of Blocks in the File:

The filesize, in blocks, is stored in this word. This size may be less than the number of blocks actually allocated to the file if the file is not a multiple of the file cluster size in length.

Creation Date:

The date the file was created is stored in RSTS internal date format.

Creation Time:

The time the file was created is stored in RSTS internal time format.

Runtime System Name:

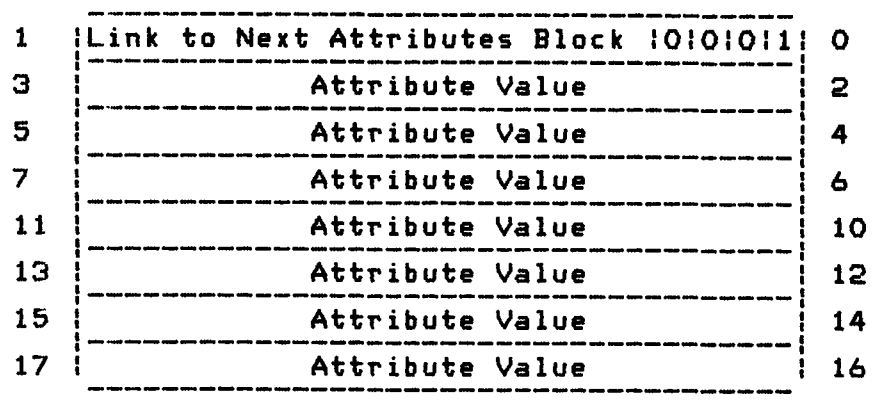
The runtime system associated with the file is stored as two words of RAD50. Unless the compile bit is set in the protection code, the RTS name is ignored.

File Cluster Size:

The file's cluster size is stored in this word.

UFD ATTRIBUTES BLOCK

Each file in the UFD can have up to 10 words of "attribute" information associated with it. This information is stored in one or two attribute blocks linked off the file's accounting block.



Link to Next Attributes Block:

If an additional attributes block is needed it is pointed to by the link in this word.

Attribute Value:

The actual value of the attributes are stored in the last seven words of the attributes block. If less than the maximum number of attributes are used, the remaining ones will be zero.

UFD RETRIEVAL INFORMATION BLOCK

Every file in the UFD that contains at least one block has some number of Retrieval Information Blocks (RIBs) associated with it. In many cases there are several RIBs for a single file, each linked together as a linked list. These RIBs provide all the information necessary to locate the physical data for a file on disk.

For each cluster of a file there is an entry in a corresponding RIB block. When a particular block on disk is desired, its physical location is computed by finding the RIB entry corresponding to the cluster which contains the block. This entry provides a pointer to the beginning of the cluster and, since clusters are contiquous, the location of the desired block can be determined by taking an offset from the beginning of the cluster.

1	Link to Next RIB	0	ULNK
3	DCN of Cluster n+0	2	UENT
5	DCN of Cluster n+1	4	
7	DCN of Cluster n+2	6	
11	DCN of Cluster n+3	10	
13	DCN of Cluster n+4	12	
15	DCN of Cluster n+5	14	
17	DCN of Cluster n+6	16	

Link to Next RIB:

This word is the link to the following RIBs. If this block is the last RIB in the list, then the link will be zero.

DCN of Cluster n+m:

The last six words of the RIB contain the device cluster numbers for the first device cluster in each file cluster. Unused words in the last RIB of the linked list will be zero.

UFD CLUSTER MAP BLOCK

Each block of the UFD contains a Cluster Map Block. It is always the last blockette of the disk block, at offset 760(8). With this cluster map it is possible to find any block of the UFD in at most one disk seek.

1	UFD Cluster Size	0
3	DCN of UFD Cluster 0	2
5	DCN of UFD Cluster 1	4
7	DCN of UFD Cluster 2	6
11	DCN of UFD Cluster 3	10
13	DCN of UFD Cluster 4	12
15	DCN of UFD Cluster 5	14
17	DCN of UFD Cluster 6	16

UFD Cluster Size:

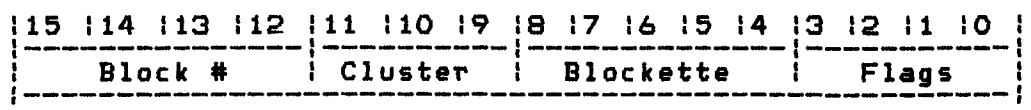
This word contains the cluster size for the UFD.

DCN for UFD Cluster n:

The last seven words of the cluster map are used to point to the physical location of each cluster of the UFD on disk.

DIRECTORY LINKS

Throughout the MFD and UFD there are many links. These links form a unique address corresponding to each blockette in the structure. Using these links and the cluster map found at the end of each block, it is possible to find any blockette in the MFD or UFD in one disk seek.



Block #:

Bits 12-15 denote the block within the cluster that is desired. Block numbers range from 0 to Cluster Size - 1.

Cluster:

Bits 9-11 denote the desired cluster. Cluster numbers range from 0 to 6.

Blockette:

Bits 4-8 denote the blockette number. Blockettes are numbered from 0 to 31.

Flags:

Bits 0-3 are used as flags. They have the following meaning:

<0> This bit denotes that the link word is in use. It is required in MFD and UFD accounting entries where the link fields may be zero.

<1> This bit is reserved for future use as a "bad block" flag. Version 7 of RSTS may provide dynamic bad block handling. If so, this bit may be used to denote a UFD or file that contains a bad block.

<2:3> These bits are currently unused but should be zero.

One note in passing is that, for some reason, the REORDR cusp sometimes sets some of these flag bits. This might cause problems if you're trying to read the MFD or UFD nonfile structured and decode these bits.

Understanding Problem Areas

If you take a little time to analyse this file structure many advantages and disadvantages become appearant. A good understanding of the way these structures relate to each other, and the implications they have on I/O, will allow you to optimize the way you do your operations.

The first important area is opening files. File opens on a nonoptimal file structure can cause an excessive amount of disk I/O looking for the desired file (worst case, over 5000 overhead disk seeks per open).

To open a file the following procedure is followed:

1. If the file is not in the current user's UFD, the linked list of name blocks in the MFD must be searched for a pointer to the root of the desired UFD. A pointer is already kept in core to the root of the current user's UFD.

2. Once the pointer to the root of the UFD is found, the UFD is searched starting at the label block. Each name block in the linked list is tested to see if it matches the desired filename.

3. Once found, the access count, status byte, and protection code are checked to see if the file may be opened in the desired manner. Assuming there are no conflicts, the file status and access count are updated and the directory block is written back to disk.

4. The contents of the first retrieval block are stored in a small buffer allocated to contain information about the open channel. This portion of the small buffer is called a "window".

Bear in mind that every time we followed one of the links down the MFD or UFD name block list we theoretically could have been required to perform a disk seek if the directory was sufficiently fragmented. I can't emphasise enough the importance of keeping your directory in good order. The REORDR cusp will optimize this correctly for many installations.

Once the file is opened the overhead can really begin. Unless the pointer to the desired cluster is already in the current window the following procedure, called a window turn, must be followed:

1. The directory block containing the name block is read into a monitor buffer called FIBUF (unless it's already there).

2. Using the pointer to the first RIB contained in the name block, the first RIB is read into FIBUF.

3. The list of RIBs is followed, reading each one in core as needed, until the desired RIB is located. The desired RIB is determined by dividing the record number by the file cluster size and dividing this result by 7. The first RIB is number 0.

4. Once the desired RIB is found, its pointers are copied to the window for future use.

5. With the new information stored in the window the desired block is retreived by determining it physical address as an offset from the beginning of the appropriate cluster.

As you can see from this discussion, using the correct cluster size for a file can make a significant difference on the amount of overhead required to get a block when it's requested. The optimal cluster size for a file is computed by dividing the total file size by 7 and taking the cluster size which is nearest the result but not smaller.

Even when you take the time to determine the optimal cluster size, it isn't always possible to fit the file within seven clusters. In this case it might be possible to make the file contiquous. If not, then at least make sure that all the RIBs for the file are located in the same block of the UFD to help optimize cacheing.

Contiquous files gaurantee that there will never be any overhead on a disk I/O request because there is never a need to get directory information once the file has been opened. The location of any block of a contiquous file can be determined by using the block number as an physical offset from the beginning of the first cluster of the file.

Extending a file can cause additional overhead on file access. Whenever a "PUT" is done to a block beyond the current end of file, a file extension takes place. A file extension causes the following procedure to be followed:

1. The SAT is searched, starting at the current block, for a contiquous free space at least as large as the desired cluster.

2. When a free space is found, the SAT is updated to show the blocks as now being in use and a pointer to the cluster is added to the appropriate RIB for the file.

3. If the "Special Extend" mode was used, the UFD is updated on disk to show the new cluster that was just added.

4. If the desired block requires that more than one cluster be added, the procedure above is repeated for each cluster.

One very important point that shouldn't be overlooked is that FIP is used whenever file opens, window turns, or file extends take place. Unlike the remainder of RSTS, FIP is a serial process which doesn't overlap any of its processing or I/O. This means that whenever a function, such as a window turn, is taking place noone else can use FIP for any purpose. Since FIP is used for almost all the SYS calls, as well as most file processing, this can have a drastic effect on system response.

With all these ways to go wrong its suprising that RSTS does as good as it does. One thing that helps a lot is disk cacheing. By now I think most everyone knows what disk cacheing is and the fantastic things it can do for you but some further understanding of how it works would be in order.

One question that seems to come up every time I discuss disk cacheing is "How much do I need". The problem is the answer is different for every installation. It depends on the number of accounts you have active at any one time, the amount of fragmentation of the UFDs, the number of different files being concurrently accessed, and the amount of memory available to user jobs.

My best suggestion is to try to understand the way disk cacheing works and then make an intelligent estimation. Try this value for a day or so under an average load and see what the result is by using Martin Minnow's STATS program (available from DECUS). Keep trying until you find a good combination. One where the Cache Hit Rate is high while the Swapping I/O, Lost Time, and FIP Waiting are as low as possible.

The way disk cacheing works is that, whenever FIP has to get a block from the UFD to do a window turn, the disk driver looks to see if the block is already contained in the disk cache area. If it is, the block is transfered from the cache to FIBUF as if a disk read had been done. If it isn't, the current contents of FIBUF are transfered to the cache, if there is an open slot, and the read is done into FIBUF.

Once a block is transfered into the cache area it remains there for at least 3 minutes. If too small a cache area is reserved then there will be very few holes to store new cache information in, thus making the cache inefficient. On the other hand, too large a cache will detract from the space available to the user area which can cause excessive swapping and increased Lost Time, which is even worse than window turns.

In summary, the following points are important for optimizing disk I/O:

1. Use the REORDR cusp or some other "directory cleanup" utility to keep the MFD and UFDs from becoming fragmented.
2. Keep the most used accounts and files near the beginning of the MFD and UFDs.
3. Optimize the cluster size of commonly used files or use contiguous files.
4. Pre-extend files wherever possible.
5. Optimize the size of the disk cache.
6. Don't open files more often than necessary.
7. Periodically rebuild your disk to keep the files as contiguous as possible.

Mike Mayfield
Data Processing Design, Inc.
6980 Aragon Circle, Suite B
Buena Park, California 90620